In the Specification

Please amend the specification of this application as follows:

Rewrite the paragraph at page 1, lines 16 to 17 as follows:

--Serial Number 09/482,902, entitled "EMULATION SUSPEND MODE WITH INSTRUCTION JAMMING" claiming priority from U.S. Provisional Application No. 60/120,809 filed February 19, 1999, now U.S. Patent No. 6,643,803;--

Rewrite the paragraph at page 1, lines 20 to 22 as follows:

--Serial Number 09/438,237, entitled "EMULATION SYSTEM WITH SEARCH AND IDENTIFICATION OF OPTIONAL EMULATION PERIPHERALS" claiming priority from U.S. Provisional Application No. 60/120,960 filed February 19, 1999, now U.S. Patent No. 6,671,665;--

Rewrite the paragraph at page 5, lines 2 to 18 as follows:

--This invention is in-circuit-emulation of an integrated circuit including a digital data processor capable of executing program instructions. A first debug event is detected during normal program execution. The causes the in-circuit-emulation to suspend program execution except for real time interrupts. A debug frame counter increments on each interrupt and ~~decements~~ decrements on each return from interrupt. If a debug event is detected during an interrupt service routine, that interrupt service routine is suspended and the count of the debug frame counter is stored. Execution of other interrupt service routines in response to corresponding interrupts is still permitted. The integrated circuit includes plural debug event detectors and the debug frame count is stored at the detector detecting a debug event during an interrupt service routine. This permits a determination of the order of interrupts triggering debug events by reading the stored debug frame count from each debug event detector.--

Rewrite the paragraph at page 11, lines 3 to 16 as follows:

--Figure 4 illustrates an electrical connection view of the coupling between access adapter 2 and target system 3. Figure 4 shows the connections of the ~~of the~~ various signals of the JTAG header 5 illustrated in Figure 2. All these signals are connected to scan controller 41. The signals nTRST, TCK and TMS are connected to two example megamodules ~~31~~ 45 and ~~33~~ 47. Figure 4 illustrates the optional connection of TCKO to the target system clock SYSCLK. The scan input TDI connects to a scan input of megamodule ~~31~~ 45. The scan output of megamodule ~~31~~ 45 supplies the scan input of megamodule ~~33~~ 47. The scan output of megamodule ~~33~~ 47 supplies the scan output TDO. The two extension signals nET0 and nET1 control megamodules ~~31~~ 45 and ~~33~~ 47 via merge unit 32. These extension signals are monitored by test equipment 43.--

C4

Rewrite the paragraph at page 14, lines 7 to 32 as follows:

--The interrupt pipeline jam for such a high priority interrupt moves the operational state to interrupt during suspend state 103. This jam causes an extra word to be pushed on the stack containing the debug status describing the reason the debug suspend state 102 entry occurred. Interrupt during suspend state 103 differs from the execute state 101 in that the interrupt processing creates a thread, linking the interrupt execution to the debug suspend state 102 as described in above. A ~~digital~~ debug frame counter (DFC) is incremented upon each high priority interrupt taken. The high priority interrupt sets an interrupt during debug state bit (IDS), which is part of the CPU status. The IDS bit sets after the context save stores the previous value on the stack with the status word. When returning from an interrupt the IDS bit indicates whether to re-enter debug suspend state 102. If the IDS bit is set, the interrupt occurred during a debug suspend state 102 and the operational state should return to the debug suspend state 102.

C5

If the IDS bit is not set, the interrupt occurred during the execute state 101 and the operational state should return to execute state 101. Upon returning from the interrupt, the PC and status return to their state before the interrupt unless the interrupt service routine has purposely modified values on the stack. This is required because it is possible for multiple interrupts to occur and be serviced while the device is in debug suspend state 102.--

Rewrite the paragraph at page 15, lines 1 to 9 as follows:

--The ~~digital~~ debug frame counter is decremented upon each return from interrupt. This count permits the debug environment to track the status of the suspended foreground task. For example, a taken high priority interrupt may change the machine state and thus the current machine state would not reflect the suspended background task. However, if the ~~digital~~ debug frame counter were zero, then the debug environment is assured no interrupts have not temporarily changed the machine state.--

Rewrite the paragraph at page 15, line 10 to page 16, line 7 as follows:

--The interrupt during suspend state 103 is exited at the end of the interrupt service routine. A normal end of an interrupt involves a return from interrupt instruction (RTI). Upon execution of a return from interrupt instruction, the machine status is popped from the stack. As noted above, the interrupt during debug state bit indicates whether the interrupt occurred during execute state 101 or debug suspend state 102. The operational state return to the former state as indicated by the interrupt during debug state bit. The prior value of the program counter is reloaded to recover the prior machine status. Execution of a return from interrupt instruction also decrements the ~~digital~~ debug frame

counter. Because it is possible to receive a higher priority interrupt while servicing a prior interrupt, more than one interrupt level may be pending. The ~~digital~~ debug frame counter indicates the current interrupt level. This is useful to debug processing as the machine status may be changed by the multiple interrupts. The debug software can read the ~~digital~~ debug frame counter and supply a debug level identity to identify currently targeted interrupt level. Execution and register operations target a specific debug level. Memory operations can target a specific debug level or bypass the level comparison. In particular, the status of the background task suspended on initial entry into debug suspend state 102 can only be reliably determined if the ~~digital~~ debug frame counter is zero. The maximum number of levels of the ~~digital~~ debug frame counter corresponds to the size of the interrupt hierarchy. This data preserves a path back to the debug suspend state 102 when the application concludes the interrupt service routine with a return from interrupt instruction.--

Rewrite the paragraph at page 16, lines 8 to 19 as follows:

--The interrupt during suspend state 103 transits to the execute state 102 upon execution of an abort interrupt (ABORTI) instruction. The abort interrupt instruction would ordinarily be used only on detection of ~~a~~ an unrecoverable error in the interrupt service routine. The path back to the debug suspend state is broken upon execution of the abort interrupt instruction. The status of the interrupt during debug state bit and the ~~digital~~ debug frame counter are ignored in this case. In particular, the interrupt during debug state bit is cleared and the ~~digital~~ debug frame counter is set to zero. This mechanism enables recovery to the background task when a high priority interrupt service routine has an unrecoverable error.--

Rewrite the paragraph at page 17, lines 5 to 18 as follows:

--Figure 7 illustrates in greater detail circuits located on each megamodule concerned with emulation. These include address comparison unit (ACU) 310, data comparison unit (DCU) 320 and the external ~~control~~ comparison unit 330 (ECU). The address comparison unit 310 provides breakpoint, counter, parallel signature analysis and data logging support. The data comparison unit 320 provides breakpoint and parallel support. The external comparison unit 330 controls external inputs to the event functions. Interaction with the programmable digital processor within the megamodule is handled by the memory unit 301. The application and debug software share access to address comparison unit 310, data comparison unit 320 and external comparison unit 330 by access to their registers.--

C9

Rewrite the paragraph at page 18, lines 9 to 22 as follows:

--The address comparison unit 310 configures for event generation where the AMSK register serves as an address mask register and the ~~a~~ AREF register serves as an address reference. The address comparison unit 310 generates a debug suspend request ~~when the ACNTL register ASTOP is TRUE~~. The AMSK field defines the address comparison unit 310 debug suspend request rudeness level. The ability to generate an event without generating a debug suspend request allows the address comparison unit 310 event to be used as a trigger generator through the ETx pins without altering core execution. This function supports breakpoints, watchpoints, and trigger generation. Table 2 shows the function specific bit mode bit definition of register ACNTL for event generation.--

C10

Rewrite the paragraph at page 27, lines 4 to 9 as follows:

--The external ~~control register~~ comparison unit 330 includes a register ECNTL that manages external events that can generate debug suspend requests. The ECNTL register manages emulation and test pin

C11

C11 zero and one inputs as well as external input used by the logic for external hardware triggering. Refer to Table 7 for a description of this function.--